Assignment 1

- Fall 2024
- Course instructor: Ashis Kumer Biswas, Ph.D.

1. If you are using python, make sure you use Python 3.10.x, and install at least these packages: `scikit-learn`, `pandas`, `numpy`, `xgboost`.
2. Also make sure you have the following data files in your workspace so that you can use them from your program. `A.csv`, `B.csv`, `C.csv`, and `scaler.joblib`. *The files can be obtained from Canvas*.
3. Enable `logging` so that all printouts from your program goes directly to a log file named `logfile.txt` which you also need to submit alongside your source code.
4. Read `A.csv` data. It's a dataset about a counterfeit bank note detection task. And, ofcourse it's a binary classification task, where `0: not a counterfeit note`, and `1: a counterfeit note`. Load data into your programming workspace.
5. Separate dependent variable `y` as the `counterfeit` column, and rest of the variables as independent variables (as `X`).
6. Split the dataset, i.e., the `(X,y)` into training (50%) and test (50%).
7. Load the scaler object from `scaler.joblib`. It's already fit to a bunch of training samples. So, don't worry about `fit` it again. Instead, you may want to use the following lines to use the already fitted `scaler` object to transform both training and test set. And, do not scale the dependent variable/feature (i.e., `y` which is the target column `counterfeit`).

```
from sklearn.preprocessing import StandardScaler
scaler = joblib.load('scaler.joblib')
X_train_scaled = scaler.transform(X_train_split)
X_test_scaled = scaler.transform(X_test_split)
```

- In case you can not load the provided `scaler.joblib` scaler object, you may want to use the mean and variance of each of the 4 features to construct the scaler object yourself. And, the formula to scale each features with Standard Scaling is simply called standardization, i.e., forcing mean of each feature to be 0 and standard deviation to be 1 with: $z = \frac{x - \mu}{\sigma}$, where μ is mean, and σ is the standard deviation, which is equal to $\sqrt{\text{variance}}$.
- Also note: you need to scale column-wise (i.e., feature-wise).

```
feature_means = [ 0.38505317,  1.9058946 ,  1.43133513, -1.21117144]
feature_variances = [ 8.01963009, 34.72221467, 18.76401568,  4.41941877]
```

8. Build your first classifier with `LogisticRegression`. Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. And, here is a suggested configuration of the model object you can use:

```
model1 = LogisticRegression(penalty='l1',
                tol=1,
                solver='liblinear',
                multi_class='auto',
                fit_intercept=False,
                max_iter=3
        )
```

- Once the `model1` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model1.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `LogisticRegression` in depth in class.

9. Build your second classifier with `KNeighborsClassifier`. Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. And, here is a suggested configuration of the model object you can use:

```
moodel2 = KNeighborsClassifier(n_neighbors=3,p=2)
```

- Once the `model2` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model2.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `K Nearest Neighbor (kNN)` classifier in depth in class.

10. Build your third classifier with `SVC` (i.e., support vector machines classifier). Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. And, here is a suggested configuration of the model object you can use:

```
model3 = SVC(gamma='auto')
```

- Once the `model3` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model3.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `Support Vector machines` classifier in depth in class.

11. Build your fourth classifier with `MLPClassifier` (i.e., Multi-layer perceptron classifier). Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. And, here is a suggested configuration of the model object you can use:

```
model4 = MLPClassifier(solver='lbfgs', alpha=0.1, hidden_layer_sizes=(1, 5))
```

- Once the `model4` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model4.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `Multi-layer perceptrons` classifier in depth in class.

12. Build your fifth classifier with `DecisionTreeClassifier`. Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. And, here is a suggested configuration of the model object you can use:

```
model5 = DecisionTreeClassifier()
```

- Once the `model5` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model5.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `DecisionTreeClassifier` in depth in class.

13. Build your sixth classifier with `RandomForestClassifier`. Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. And, here is a suggested configuration of the model object you can use:

```
model6 = RandomForestClassifier(max_depth=2)
```

- Once the `model6` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model6.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `RandomForestClassifier` in depth in class.

14. Build your seventh classifier with `XGBClassifier`. Here is the documentation you may want to read/review: https://xgboost.readthedocs.io/en/stable/get_started.html. And, here is a suggested configuration of the model object you can use:

```
model7 = XGBClassifier(n_estimators=2, max_depth=2, learning_rate=1,
objective='binary:logistic')
```

- Once the `model7` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model7.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `XGBoost` classifier in depth in class.

15. Build your eighth classifier with `GaussianNB` (Gaussian Naive Bayes classifier). Here is the documentation you may want to read/review: https://scikit-learn.org/stable/modules/naive_bayes.html. And, here is a suggested configuration of the model object you can use:

```
model8 = GaussianNB()
```

- Once the `model8` object is instantiated, call the `fit` method on the training split.
- Make sure to save the model with `joblib` library and as a file named `model8.joblib`. Please do not submit the joblib file. It's only for your later usage of this model.
- Please note: we will study `GaussianNB` classifier in depth in class.

16. **Get predictions for the 2 samples in `B.csv`**:
    - Have each of the 8 models predict the 2 samples present in the `B.csv` file. Please note: the true annotations for the two samples are not present in the file. Consider this situation like you are applying your model in real-world use. How cool is that!
    - Make sure you print the model name (i.e., something like `model1`) and the predicted class labels for each of the two samples.
    - Also, print the consensus predictions for the two samples (with some type of majority voting and such).
    - Don't forget to scale the features of the samples using the `scaler` object provided before sending to the models for prediction.
17. **Evaluate models based on `C.csv` dataset**:
    - `C.csv` contains annoted samples. Use the target annotations to evaluate all the 8 models. Please report `accuracy`, `precision`, `recall`, `f1-score`, `false-discovery rate`, `matthews correlation coefficient`
    - Please don't forget to scale the samples present in this dataset prior to sending to the models for prediction.
    - Also, comment which model performs the best in each of the 6 evaluation metrics.
    - Also, comment which model do you think is `acceptable` to deploy in a real bank. Justify your reasoning. Also, avoid answers like `none is acceptable`. 😃